Dynamic BIM model conversion as inference-based ontology alignment*

Pierre Bourreau[1] and Jyrki Oraskari[2]

[1] Nobatek/INEF4, 9 rue Jean-Paul Alaux, 33000 Bordeaux, France
pbourreau@nobatek.inef4.com
[2] CAAD RWTH Aachen, Templergraben 55, 52062 Aachen, Germany
Jyrki.Oraskari@dc.rwth-aachen.de

Abstract. The AEC sector is known to be highly fragmented, different experts requiring different information. Current BIM collaborative practices can be described as static as they are based on file exchange, mainly using IFC files. Instead, in an ideal level 3 BIM platform, different stakeholders could work together on a single centralized building model, using different domain-specific vocabularies, and updating the model synchronously. We propose an approach based on semantic rule reasoning to dynamically convert information in different vocabularies. While existing alignments only cover class conversions, we develop inference rules that handle conversion of relations in a dynamic and transparent way, being executed by a reasoner. The approach is implemented and tested on converting models from ifcOWL to BIM4Ren, a BOT-based ontology. The scalability of the implementation is discussed as well as its limitation.

Keywords: BIM, alignment, logic programming, interoperability

1 Introduction

The fragmentation of the AEC industry is a well-known issue in construction projects: different AEC (Architecture and Engineering in Construction) actors use different vocabularies and information to describe the same building. BIM as a collaborative process supported by digital tool and models intends to ease data exchange between project stakeholders. To do so, the industry delivered IFC or the ISO 10303-21 IFC-SPF file format, which aims at being a standard exchange model to which all software vendors can comply, at the cost of a translation mechanism. We call such a data exchange 'static' as each transferred model is a picture of the model at some instant; indeed, if multiple actors are allowed to update it, some file merging functionalities become necessary. In an ideal BIM level 3 environment, as mentioned in the literature, all actors work on a single and centralized model, and updates are handled automatically so that each actor

* The BIM4Ren project has received funding from the European Union's Horizon H2020 research and innovation programme under grant agreement No 820773

works on the latest model version. In the present article we investigate a linked data-based approach that reduces model conversion to an ontology mapping problem, for a model stored in a single triple store. With such approach, different users may use different vocabularies to work on the very same centralized model. It could be used for 'horizontal' (i.e. from one model to another, like IFC2x3 to gbXML) or 'vertical' conversion (i.e. from one model version to another one, like IFC2x3 to IFC4)

Ontology mapping can be seen as a task where concepts (i.e. entities and relations) of different vocabularies are mapped based on their semantic correspondence and organisation, keeping the intended interpretations [9,10]. In the Linked Building Data domain, a few ontology alignments exist [16]. While many ontology mapping techniques exist, the article focuses on logic-based mapping by elaborating inference rules that are interpreted by a reasoner that can dynamically generate matching relations.

The article is structured as follows: section 2 introduces the problem of collaboration and model conversion in the BIM community, and ontology mapping. We also introduce ifcOWL and the BIM4Ren ontology as the two ontologies used to test our approach. Section 3 presents different ontology mapping strategies used to align the two ontologies above. In section 4, we introduce the implementation, test it, discuss issues and solutions proposed, and compare the approach with a programmatic one.

2 BIM collaboration and ontology mapping

2.1 BIM collaboration

BIM is the main asset towards a digitization of the AEC community. The sector being highly fragmented, data exchange between project stakeholders is an issue that BIM collaborative process supported by BIM models intends to solve. In this direction, BIM levels are often used to describe the collaboration maturity of a project: while almost no collaboration are required in a BIM level 0 or 1, BIM level 2 requires project stakeholders to exchange information through files. In an ideal BIM level 3, all actors work on the very same model, without any need of exchanging files.

At the moment, the community seems to focus its efforts on improving BIM level 2, through the delivering of different BIM standards: IFC (Industry Foundation Classes) is the main one; it is rich, extensible and covers most AEC domains. But its complexity seems to be a barrier to software vendors, and reaching an agreement on the vocabulary and the required information that needs to be in such digital model is highly challenging, considering different countries or regions may have different requirements, and even projects can have their own specificities [2]. COBie is another lightweight building model, that focuses on non-geometrical information; in energy modelling gbXML is a *de facto* standard for building energy models. More recently, knowledge graph-based modelling gained some interests as a new paradigm to deliver BIM models: [1,12] proposed ifcOWL as an OWL translation of the IFC EXPRESS schema format;

[15] proposed BOT (Building Ontology Topology) as a minimal knowledge representation for any building project.

It therefore seems that while the community is aware of the need for stakeholders to share information, more and more data models are created. This could be contradictory with BIM level 3 principle where a single model, and therefore a single vocabulary is potentially used, unless some automatic conversions are implemented. While many such converters exist, they are all written in some specific programming language (Java, Python . . .) and their implementation into a single BIM repository does not seem trivial.

2.2 Ontology mapping

In linked data, a knowledge graph is a set of statements in which nodes and edges have well-defined meanings. Linking knowledge graphs, known in the literature as *ontology mapping* or ontology matching, is done to benefit from the work of a specialized community; the resulting ontology linkage is a set of relations between concepts of different ontologies, known as an *ontology alignment* [5]. Ontology mappings can be done in different ways: for instance word distance, translation from one language to another, or graph pattern similarities, can be used to discover similarities between graphs.

The present work focuses on semantic matching techniques [17,7], which relies on logical inference to deduce alignments between models. This is done defining:

1. (**Axioms**): direct ontology alignments between classes/relations; those take the form $\vdash$ `Ax` where `Ax` is a list of triples (i.e. usually called literals in formal logic) `<s; p; o>` where `s` belongs to an ontology $\mathtt{O}_1$, `o` to an ontology $\mathtt{O}_2$ and p is an alignment predicate.
2. (**Inference rules**) create logical rules that can be triggered thanks to the creation of new alignments; those are of the form (`A` $\vdash$ `C`, where `A` and `C` are list of triples respectively called antecedents and consequents)

Triples in inference rules are usually *open* triples, i.e. they contain variables. Nevertheless, only *safe rules* can be triggered correctly; a rule is said to be safe when every variable in its consequent is already declared in its antecedent.

As an example, the transitive closure of the `rdfs:subClassOf` relation can be achieved with the two following rules: (i) states that an instance of `owl:Class` is a subclass of itself (i.e. reflexivitiy); and (ii) that a `rdfs:subClassOf` of a class y, is a subclass of all super classes of y (i.e. transitivity).

$$(i)\ \texttt{(?x rdfs:type owl:Class)} \vdash \texttt{(?x ex:transSubClass ?x)}$$

$$(ii)\ \texttt{(?x rdfs:subClassOf ?y) (?y ex:transSubClass ?z)}$$
$$\vdash \texttt{(?x ex:transSubClass ?z)}$$

Finally, reasoners can evaluate rules in different ways: (i) a *forward-chaining* reasoning considers every triple in the ABox that instantiates the antecedent of a rule to generate new triples instantiating the consequent; the operation is repeated until a goal statement is reached, or no more triples are added to the ABox. (ii) a *backward-chaining* approach triggers rules on request to satisfy the instantiation of the consequent, through an instantiation of its antecedents. It is known that a forward-chaining avoids iterating in infinite loop, but is memory-consuming. On the other hand, backward-chaining can run in infinite loops if no memoization is turned on, but does not store unnecessary information. Combining both forward and backward-chaining evaluable rules is called *hybrid* reasoning.

2.3 ifcOWL and BIM4Ren

The present work focuses on converting ifcOWL models into the BIM4Ren vocabulary. IfcOWL [1,12] is the direct translation of the EXPRESS-based IFC schema into the OWL language; its coverage is therefore as wide as the IFC one. The BIM4Ren data model [3] embodies a set of ontologies with the aim of modelling energy-efficient renovations. It resides on three independent layers (core, product and domain) and a transversal one (utility) as represented in Figure 1, which makes it a product-centric model in the sense that: (i) the core layer is made of ontologies that help relate products between them or inside a specific building context, (ii) the product layer is a taxonomy used to assign a specific class to a product, and (iii) the domain layer is used to associate properties to a product; according to specific domains.

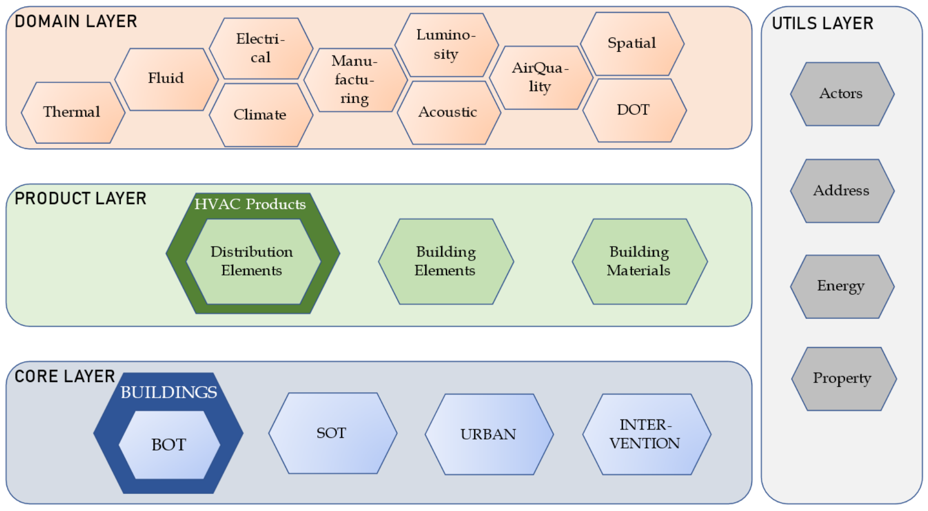


Fig. 1. The BIM4Ren data model architecture

Full modularity (i.e. sub-ontologies are pairwise independent: none needs importing another BIM4Ren ontology but from the Utils layer) is achieved through a multiple instantiation mechanism, that allow importing modules on demand. This enables one to use only small models of interest for a specific task instead of a complex multi-domain model.

As an example, a boiler `_:boiler1` may be modelled as an instance of `bot:BuildingElement` to locate it in the building; of `sot:SystemElement` to describe its role in the MEP networks; of `ifc4:Boiler-WATER` in the distribution elements model; and of `elec:Consumer` in the electrical ontology, where `elec:Consumer` comes with a `elec:nominalPower` relation.

At the technical level, the BIM4Ren ontology uses (i) an extension of BOT [15] to describe buildings, with reference to Omniclass classification to assign functional properties to buildings (i.e. apartment, allotment,. . .) and spaces (kitchen, bedroom,. . .); (ii) the product layer is made of taxonomies extracted from ifcOWL to classify distribution[3] and building elements[4], where enumerated product types are transformed into classes; (iii) and the property layer uses SML, the property modelling on-going standard provided by the CEN[4], which is based on QUDT [6] and Ontology for Property Management (OPM) level 2 [14].

3 Model conversion as inference

In the present section, a rule-based ifcOWL to BIM4Ren conversion is presented. As mentioned in section 2.2, we first identify the relations used to align the two models (i.e. the axioms) and then detail the inference rules that are used.

3.1 Axioms

Different vocabularies can be used to directly map classes or relations of different models. The `rdfs:subClassOf` or `owl:equivalentClass` relations are used in the current implementation to map classes, in the same way as it is done in the existing alignment between BOT and ifcOWL [5]. Extending this approach to the SOT ontology leads to the following set of rules:

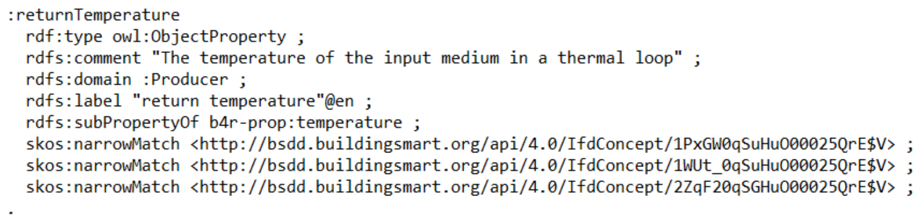
```
ifc:IfcDistributionSystem rdfs:subClassOf sot:System.
ifc:IfcDistributionElement rdfs:subClassOf sot:SystemComponent.
ifc:IfcDistributionPort rdfs:subClassOf sot:TransportElement.
ifc:IfcFlowTerminal rdfs:subClassOf sot:TerminalElement.
```

These rules form the axioms of our conversion process from ifcOWL to BIM4Ren. Nevertheless, it is worth mentioning that other vocabularies are used to map the BIM4Ren models with other vocabularies, in particular Omniclass and Property Set Definition (PSD).

[3] https://pi.pauwel.be/voc/distributionelement/index-en.html
[4] https://pi.pauwel.be/voc/buildingelement/index-en.html
[5] https://raw.githubusercontent.com/w3c-lbd-cg/bot/master/IFCOWL4_ADD2Alignment.ttl

As mentioned in 2.3, Omniclass is used to extend BOT with concepts on building and space types. Because to our knowledge no web referenceable version of Omniclass exists, every created subclass is bound to its corresponding associated Omniclass code through the `rdfs:isDefinedBy` relation (e.g. `buildings:Bathroom rdfs:isDefinedBy "Omniclass 13-65 13 15"`.

In addition, properties of the domain layer are mapped with their PSD counterpart, which are web referenceable through the buildingSmart Data Dictionary (bsDD). A BIM4Ren property may be associated to multiple bsDD properties (see Figure 2), using the `skos` vocabulary.



```
:returnTemperature
  rdf:type owl:ObjectProperty ;
  rdfs:comment "The temperature of the input medium in a thermal loop" ;
  rdfs:domain :Producer ;
  rdfs:label "return temperature"@en ;
  rdfs:subPropertyOf b4r-prop:temperature ;
  skos:narrowMatch <http://bsdd.buildingsmart.org/api/4.0/IfdConcept/1PxGW0qSuHuO00025QrE$V> ;
  skos:narrowMatch <http://bsdd.buildingsmart.org/api/4.0/IfdConcept/1WUt_0qSuHuO00025QrE$V> ;
  skos:narrowMatch <http://bsdd.buildingsmart.org/api/4.0/IfdConcept/2ZqF20qSGHuO00025QrE$V> ;
.
```

Fig. 2. Example of alignment with IFD through bsDD

3.2 Inference rules

The presented axioms are limited to mapping ifcOWL classes to BIM4Ren, as a direct mapping (using `rdfs:subPropertyOf` for instance) is impossible. Indeed, while relations in ifcOWL are often objectified (i.e. a class is used to model them, with the `IfcRel` prefix), linked building data ontologies simplify modelling using a single relation. This is illustrated in Figure 3.

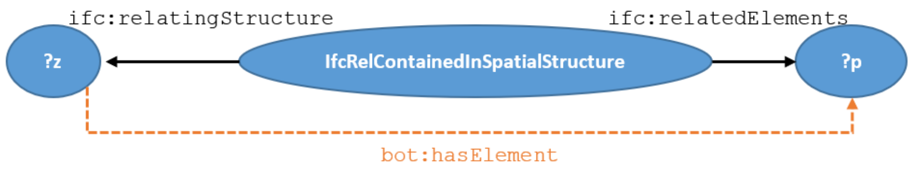


Fig. 3. Example of deobjectification of relations

Deobjectifying a relation can easily be translated as an inference rule:
`(?z rdf:type ?cl) (?cl rdfs:subClassOf bot:Zone)`
`(?c ifc:relatingStructure_IfcRelContainedInSpatialStructure ?z)`
`(?c ifc:relatedElements_IfcRelContainedInSpatialStructure ?p)`
$\vdash$ (?z `bot : hasElement` ?p)

Some complementary internal rules need to be elaborated to deal with the list structure of ifcOWL. For instance the range of the `IfcRelNests_relatedObjects` relation is a list, which type in ifcOWL is `https://w3id.org/list`; it is a LISP-like list structure made of `list:hasContents` to get the head, and `list:hasNext` to return the list from which the top element was removed [13]. Handling lists in the converter requires to have a way to directly check whether an element is in a list, which can be done with the following recursive rules:

```
(?l list:hasContents ?elt) ⊢ (?elt list:isIn ?l)
(?l list:hasNext ?q) (?elt list:isIn ?q) ⊢ (?elt list:isIn ?l)
```

Finally, classes in the Product layer of the BIM4Ren data model also need to be aligned with the corresponding enumerated types in IFC. For each enumerated type, a conversion rule can be elaborated as below:

```
(?z rdf:type ifc:IfcBoiler) (?z ifc:predefinedType_Boiler ifc:WATER)
⊢ (?z rdf:type distribution:Boiler-WATER).
```

In order to deal with all enumerated types in the Distribution and Building element ontologies, a Python script was developed. This leads to generating 693 rules amongst the 708 total rules.

4 Implementation and tests

In order to evaluate our approach, we compare it with a program-based converter considering (i) correctness of the conversion, (ii) performance, (iii) maintenance, (iv) the full conversion workflow.

4.1 Description

An implementation of our approach was done on an Apache Jena instance. All rules were implemented according to the Generic Rule Reasoner file format. The program-based converter to which results were compared is a Java-based converter[6], an extension of the IFCtoLBD [11] converter. Regarding the inference engine approach, we tested two different reasonings: a simple forward-chaining and a hybrid one, which differs from the former in that:

(i) It uses an ad hoc `subClass` relation: the inference rules need to check that some individuals are instances of specific classes, or of descendant of such classes. The `b4r:subClassOf`) is the transitive closure of `rdfs:subClassOf` but it is restricted to ancestor classes used in the inference rules (i.e. `bot:Zone`, `sot:System` and `sot:SystemComponent`).

(ii) It uses backward-chaining on complex rules: all rules used to align the core layer are converted to top-down rules. This can be simply done in Jena's reasoner[7].

[6] `https://github.com/jyrkioraskari/IFCtoB4R-DM_OpenAPI/`

[7] `https://jena.apache.org/documentation/inference/index.html#RULEbackward`

(iii) It memorizes recurrent inferred triples: Some of the rules do not directly solve alignment issues but can be seen as utility rules. This is the case of the `b4r:subClassOf` and `list:isIn` rules. Because these rules can be used extensively inside other rules and because they potentially generate a high number of triples, a memorization mechanism is enable to allow keeping results in memory and to avoid infinite loops.

Two different IFC models were considered for testing. The first one (called Dunant.ifc) results from the scanning of a multi-dwelling building in Paris as part of the BIM4Ren project. The corresponding ifcOWL ttl file is made of 10033 individuals, from which only 138 are not related to geometry; those are instances of `IfcSite`, `IfcBuilding`, `IfcStorey`, `IfcSlab`, `IfcCovering`, `IfcWall`, `IfcDoor` and `IfcWindow`. To test how scalable the approach is, the Duplex_MEP model [8] was considered. Duplex is two-storeys appartement model, well-known in the IFC community[8]; it is made of 687363 individuals (apart from geometric information) and more importantly for our experiments, contains instances of `IfcSpace` of the `DistributionElement` hierarchy, so most of the inference rules can be tested. Those rules were created to infer the `hasBuilding`, `hasStorey`, `hasSpace` and `hasElement` relations in BOT, and the `hasElements` (i.e. when an element is a component of a system) and `connectedWith` (i.e. when two MEP products are connected) relations in SOT.

Table 1. Summary of expected relations - generated with IFC2BOT

Relations	**Dunant**	**Duplex_MEP**
`bot:hasStorey`	2	3
`bot:hasSpace`	0	42
`bot:hasElement`	59	926
`bot:containsElement`	0	167

Regarding the methodology, (i) we ensure the correctness of the rules through a comparison of the BOT classes and relations output by our converters (obtained through SPARQL query) with the ones output by the IFC2BOT [9]; (ii) we compare the execution performance of the inference engine approach with the one of the programmatic approach; to do so, we perform queries on a triple store for which an inference engine is running, and on another one on which the model output by the IFCtoLBD tool was uploaded.

4.2 Results

First, the tests prove some scalability issues for the forward-chaining approach: Jena stopped with a "java.lang.OutOfMemoryError: Java heap space" error, after the execution hang for more than 30 minutes on the first query performed on

[8] available at `https://github.com/buildingSMART/Sample-Test-Files/tree/master/IFC%202x3/Duplex%20Apartment`

[9] `https://github.com/NIRAS-MHRA/IFC2BOT`

the Duple_MEP model. As already mentioned, forward-chaining reasoning may require keeping in memory useless triples, and by default the GenericRuleReasoner infers all new triples on the first query performed. The hybrid reasoner runs successfully on all models, and all queries, proving the efficiency of the different modifications performed.

Regarding the correctness of the approach, the hybrid reasoning's BOT outputs are sound and complete with respect to the IFC2BOT outputs, on both the Dunant and Duplex_MEP files, ensuring its correctness. The same happens for the forward-chaining reasoning on the Dunant file.

Regarding execution performance, the results are described in Table 2. Operations were run in the same order given as in the table.

Table 2. Performance test results (times in ms)

Operation	**Forward-chaining**		**Hybrid**		**IFCtoLBD**	
	Dunant	Duplex	Dunant	Duplex	Dunant	Duplex
1.Get all `bot:Building`	4138	Err	159	225	725	3902
2.Get all `bot:Space`	766	Err	157	150	645	4338
3.Get all `bot:Zone`	745	Err	138	10	915	4406
4.Get all `bot:containsZone`	3	Err	20	39	938	3607
5.Get all `bot:hasElement`	4	Err	5	92	977	4494
6.Get all `sot:SystemComponent`	732	Err	137	150	-	-

First of all, for the forward-chaining reasoning, getting all buildings takes longer than getting all spaces or zones, but this is due to the testing procedure: getting buildings was the first query performed, and it therefore triggers the reasoner that infers all new triples, a time-consuming operation. In fact when repeating the query, answers are provided within approximately 700ms.

The hybrid evaluation performs better than the forward-chaining one on queries 1, 2 and 3 on Dunant. The performed queries are of the form: `SELECT ?z WHERE {?z a ?cl. ?cl REL ENT.}`, where `ENT` is to be replaced with the goal class (`bot:Building` on 1., `bot:Space` on 2. and `bot:Zone` on 3.) and `REL` with `rdfs:subClassOf*` (i.e. the transitive closure of `rdfs:subClassOf`) for the forward-chaining evaluation, and with `b4r:subClassOf` for the hybrid one. This gives an idea of the efficiency provided by adding the former over the latter. To confirm it, getting all buildings on Duplex in the hybrid approach with the `rdfs:subClassOf*` relation takes 22 533ms.

Generally speaking, the hybrid reasoning outperforms the forward-chaining one for almost all queries on the Dunant file; this tends to show there is no clear advantages on computing all inferred triples at initialization. This is surprisingly clear with queries 4 and 5, where the relations are inferred with the help on non-axiomatic rules and may therefore require high runtime computation in the hybrid implementation. Moreover, the more different queries are run on the hybrid model, the better it performs. Indeed, memorization of the results (either intermediate or final) of multiple queries ensures those can be reused without any computational cost for future queries.

In addition, queries on a logic-based approach are competitive with the Java program, if not outperforming it, in the case of the hybrid implementation. The high difference between the hybrid approach and IFCtoLBD seems again to reside on the use of the `b4r:subClassOf` relation.

Regarding maintenance, the inference rule-based approach may not be as easy to develop or make evolve as a program-based approach. Indeed, debugging inference rules in Jena using a configuration file for the rules is not trivial, and, to our knowledge, no tool exists to assist developers in this task as opposed to debugging environments for long-time consolidated programming languages. Moreover, the syntax for the inference rules may differ from a triple store to another one.

Finally, regarding the workflow, the inference rule-based approach resides in using a triple store as a central model storage, and its native engine as a converter; no deployment effort is therefore needed as all required functionalities are available within Jena or any other triple store. The program-based conversion requires (i) to download the ifcOWL model, (ii) convert it to the BIM4Ren model, (iii) upload it again to the triple store. This process can probably be automatized, but the advantage in using an inference-rule based approach is that every changes made, using IFC (resp. BIM4Ren) vocabulary, is automatically reflected in the BIM4Ren (resp. IFC) one.

5 Conclusion

5.1 Advantages and limitations

The experiments show that the linked-data approach is efficiently dealing with the conversion of models from IFC to BIM4Ren. Nevertheless, the approach is scalable at the cost of some tuning on the rules evaluation. The hybrid approach differs from the forward-chaining one as the latter converts parts of the model on-demand, while the former maintains the conversion at run time. Nevertheless, both dynamically convert only the necessary part of the model, which is the main difference with a programmatic approach, that converts the the full model at once. A programmatic approach may have a wider coverage, being able to deal with geometry conversion as well as properties, but our approach is language-independent, can be directly implemented on any triple store and is dynamic.

The current work also led to some diagnosis on ifcOWL. As mentioned, we created specific rules for the `isIn` relation, which checks the presence of an element inside an `owl:OWLList`. Using `rdf:List` instead, the built-in `swrlb:member` could be used[10], which may improve the overall performance of the converter.

Finally, some experiments were done on backward conversions, i.e. from BIM4Ren to ifcOWL. The main issue is that backward inference rules are unsafe, because ifcOWL uses objectified relations, introducing therefore more concepts

[10] see 8.7 of `https://www.w3.org/Submission/SWRL/`

than BIM4Ren. In practice, this can be bypassed using the `makeTemp` Jena built-in function. This is illustrated in the rule above where the variable `?c` is the objectified relation to create (i.e. the main element in the consequent), and as such, it needs to be declared in the antecedent.

```
(?z rdf:type ?clz) (?clz b4r:subClassOf ifc:IfcSpatialElement)
(?z bot:hasElement ?p)
(?p rdf:type ?clsp) (?clsp b4r:subClassOf ifc:IfcProduct)
makeTemp(?c)
-> (?c rdf:type ifc:IfcRelCointainedInSpatialStructure)
(?c ifc:relatingStructure_IfcRelContainedInSpatialStructure ?z)
(?c ifc:relatedElements_IfcRelContainedInSpatialStructure ?p)
```

5.2 Future work

In the future, the approach should be extended to cover PSD properties and geometry. Regarding properties, a mapping will be done in the BIM4Ren project between the model and a POBIM (ISO 12006-3) implementation, that is used to describe product properties. If a formal ontology for PSD properties arises, it will also be used to convert such properties into relations in the Domain layer of the BIM4Ren model.

Extending the approach to geometry would require a deeper work, because the amount of geometry information in BIM is high, and the scalability of such approach cannot be easily ensured. In addition, dealing with geometry in a linked data environment is still a reasearch topic in itself to the authors knowledge. Ideally, such conversion should also imply geometry representation conversion, i.e. bounding boxes, B-REP. . .

Extending the approach to models used in the industry, in particular gbXML or COBie, is another interesting topic as it would deliver a more impacting use case to the overall AEC industry.

Other alignment methodologies may also be explored to solve BIM issues in the AEC community; for instance the rule-based alignment is manually done through the elaboration of the rules, but other approaches can be tested to discover alignments automatically (patterns discovery, word distance...).

Overall, in a BIM level 3 implementation, our approach could be used to allow stakeholders to use their own vocabularies, which we consider crucial to federate the sector. Such an approach strongly differs from bringing a single unified standard. Other tests could be performed to ensure the scalability of the solution on bigger models, concurrency issues on triple stores when updating the model,. . .

References

1. Beetz, J., van Leeuwen, J., de Vries, B.: IfcOWL: A case of transforming EXPRESS schemas into ontologies. AI EDAM **23**(1), 89–101 (2009)

2. van Berlo, L.: BuildingSMART tehnical roadmap (2020), published on buildingsmart.org, version 30 april 2020
3. Bourreau, P., Charbel, N., Werbrouck, J., Senthilvel, M., Pauwels, P., Beetz, J.: Multiple inheritance for a modular BIM. In: Teulier, R., Marquès, S. (eds.) Le BIM et l'évolution des pratiques: Ingénierie et architecture, enseignement et recherche. pp. 63–82. Eyrolles (2020)
4. CEN: prEN 17632:2021, Semantic Modelling and Linking (SML) (2020), (out for first ballot in March 2021)
5. Euzenat, J., Shvaiko, P., et al.: Ontology matching, vol. 18. Springer (2007)
6. Hodgson, R., Keller, P.J., Hodges, J., Spivak, J.: QUDT-quantities, units, dimensions and data types ontologies. USA Available http://qudt. org March **156** (2014)
7. Jean-Mary, Y.R., Shironoshita, E.P., Kabuka, M.R.: Ontology matching with semantic verification. Journal of Web Semantics **7**(3), 235–251 (2009)
8. Johnson, M., Fallon, K.K.: Experimental building information models. Tech. rep., Construction Engineering Research Laboratory (US) (2011)
9. Kalfoglou, Y., Schorlemmer, M.: Ontology mapping: the state of the art. In: Dagstuhl Seminar Proceedings. Schloss Dagstuhl-Leibniz-Zentrum für Informatik (2005)
10. Mao, M.: Ontology mapping: An information retrieval and interactive activation network based approach. In: The Semantic Web, pp. 931–935. Springer (2007)
11. Oraskari, J., Bonduel, M., McGlinn, K., Pauwels, P., Priyatna, F., Wagner, A., Kukkonen, V., Steyskaland, S., Lehtonen, J.: jyrkioraskari/IFCtoLBD: IFCtoLBD 2.5 (Sep 2020). https://doi.org/10.5281/zenodo.4056940, `https://doi.org/10.5281/zenodo.4056940`
12. Pauwels, P., Terkaj, W.: EXPRESS to OWL for construction industry: Towards a recommendable and usable ifcowl ontology. Automation in Construction **63**, 100–133 (2016)
13. Pauwels, P., Terkaj, W., Krijnen, T., Beetz, J.: Coping with lists in the ifcOWL ontology. In: 22nd EG-ICE International Workshop. pp. 113–122 (2015)
14. Rasmussen, M., Lefrançois, M., Bonduel, M., Hviid, C., Karlshøj, J.: OPM: An ontology for describing properties that evolve over time. In: 6th Linked Data in Architecture and Construction Workshop (2018)
15. Rasmussen, M.H., Lefrançois, M., Schneider, G.F., Pauwels, P.: BOT: the building topology ontology of the w3c linked building data group. Semantic Web **12**, 1–19 (2019)
16. Schneider, G.F.: Towards aligning domain ontologies with the building topology ontology. In: Proceedings of the 5th Linked Data in Architecture and Construction Workshop (LDAC 2017) (2017)
17. Shvaiko, P.: Iterative schema-based semantic matching. Ph.D. thesis, University of Trento (2006)

A Axioms

```
@prefix : <https://models.bim4ren.eu/alignment/buildings-IFC#> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix dcterms: <http://purl.org/dc/terms/> .
@prefix bot: <https://w3id.org/bot#> .
@prefix sot: <https://models.bim4ren.eu/sot#> .
@prefix ifc4: <https://standards.buildingsmart.org/IFC/DEV/IFC4_1/OWL#> .
@prefix ifc2x3: <https://standards.buildingsmart.org/IFC/DEV/IFC2x3/TC1/OWL#> .

bot:Site owl:equivalentClass ifc4:IfcSite ;
           owl:equivalentClass ifc2x3:IfcSite.
bot:Building owl:equivalentClass ifc4:IfcBuilding ;
                 owl:equivalentClass ifc2x3:IfcBuilding .
bot:Storey owl:equivalentClass ifc4:IfcBuildingStorey  ;
                 owl:equivalentClass ifc2x3:IfcBuildingStorey .
bot:Space owl:equivalentClass ifc4:IfcSpace ;
            owl:equivalentClass ifc2x3:IfcSpace .
bot:Element owl:equivalentClass ifc4:IfcElement ;
            owl:equivalentClass ifc2x3:IfcElement .

ifc2x3:IfcDistributionSystem rdfs:subClassOf sot:System.
ifc4:IfcDistributionSystem rdfs:subClassOf sot:System.

ifc2x3:IfcDistributionElement rdfs:subClassOf sot:SystemComponent.
ifc4:IfcDistributionElement rdfs:subClassOf sot:SystemComponent.

ifc2x3:IfcDistributionPort rdfs:subClassOf sot:TransportElement.
ifc4:IfcDistributionPort rdfs:subClassOf sot:TransportElement.

ifc2x3:IfcFlowTerminal rdfs:subClassOf sot:TerminalElement.
ifc4:IfcFlowTerminal rdfs:subClassOf sot:TerminalElement.
```

B Extract of the hybrid rules used in the experiments

```
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix list: <https://w3id.org/list#> .
@prefix bot: <https://w3id.org/bot#> .
@prefix sot: <https://models.bim4ren.eu/sot#> .
@prefix b4r-building: <https://models.bim4ren.eu/buildings#> .
@prefix distribution: <https://pi.pauwel.be/voc/distributionelement#>.
@prefix building: <https://pi.pauwel.be/voc/buildingelement#>.
@prefix b4r: <https://models.bim4ren.eu#> .
@prefix ifc: <https://standards.buildingsmart.org/IFC/DEV/IFC2x3/TC1/OWL#> .

[equivalent1: (?a owl:equivalentClass ?b) -> (?a rdfs:subClassOf ?b) (?b
rdfs:subClassOf ?a)]

-> table(b4r:subClassOf).
[transSubClassZone:  -> (bot:Zone b4r:subClassOf bot:Zone)]
[transSubClassSystemComponent:  -> (sot:SystemComponent b4r:subClassOf
sot:SystemComponent)]
[transSubClassSystem:  -> (sot:System b4r:subClassOf sot:System)]
[transSubClass1: (?cls1 b4r:subClassOf ?cls2) <- (?cls1 rdfs:subClassOf ?cls2)]
[transSubClass2: (?cls1 b4r:subClassOf ?cls3) <- (?cls1 rdfs:subClassOf ?cls2)
(?cls2 b4r:subClassOf ?cls3)]

-> table(list:isIn).
[isIn1: (?elt list:isIn ?l) <- (?l list:hasContents ?elt)]
[isIn2: (?elt list:isIn ?l) <- (?l list:hasNext ?queue) (?elt list:isIn ?queue)]

[botHasElement-IFC: (?z bot:hasElement ?p) <-
 (?c ifc:relatingStructure_IfcRelContainedInSpatialStructure ?z)
 (?c ifc:relatedElements_IfcRelContainedInSpatialStructure ?p)
 (?z rdf:type ?cl) (?cl b4r:subClassOf bot:Zone)]

[relHasBuilding-IFC2x3: (?z1 bot:hasBuilding ?z2) <-
 (?rel ifc2x3:relatingObject_IfcRelDecomposes ?z1)
 (?rel ifc2x3:relatedObjects_IfcRelDecomposes ?z2)
 (?z1 rdf:type ?cl1) (?cl1 b4r:subClassOf bot:Zone)
 (?z2 rdf:type ?cl2) (?cl2 b4r:subClassOf bot:Building)]
[relHasStorey-IFC2x3: (?z1 bot:hasStorey ?z2) <-
 (?rel ifc2x3:relatingObject_IfcRelDecomposes ?z1)
 (?rel ifc2x3:relatedObjects_IfcRelDecomposes ?z2)
 (?z1 rdf:type ?cl1) (?cl1 b4r:subClassOf bot:Zone)
 (?z2 rdf:type ?cl2) (?cl2 b4r:subClassOf bot:Storey)]
[relHasSpace-IFC2x3: (?z1 bot:hasSpace ?z2) <-
 (?rel ifc2x3:relatingObject_IfcRelDecomposes ?z1)
 (?rel ifc2x3:relatedObjects_IfcRelDecomposes ?z2)
 (?z1 rdf:type ?cl1) (?cl1 b4r:subClassOf bot:Zone)
 (?z2 rdf:type ?cl2) (?cl2 b4r:subClassOf bot:Space)]

[relContainsZone-IFC4: (?z1 bot:containsZone ?z2) <-
 (?rel ifc4:relatingObject_IfcRelAggregates ?z1)
 (?rel ifc4:relatedObjects_IfcRelAggregates ?z2)
 (?z1 rdf:type ?cl1) (?cl1 b4r:subClassOf bot:Zone)
 (?z2 rdf:type ?cl2) (?cl2 b4r:subClassOf bot:Zone)]


[connectedWith-IFC: (?elt1 sot:connectedWith ?elt2) <-
 (?c ifc:relatingPort_IfcRelConnectsPorts ?p1) (?c
ifc:relatedPort_IfcRelConnectsPorts ?p2)
 (?elt1 rdf:type ?cls1) (?cls1 b4r:subClassOf sot:SystemComponent)
 (?elt2 rdf:type ?cls2) (?cls2 b4r:subClassOf sot:SystemComponent)
 (?n1 ifc:relatingObject_IfcRelNests ?elt1) (?n1 ifc:relatedObjects_IfcRelNests
?list1) (?p1 list:isIn ?list1)
 (?n2 ifc:relatingObject_IfcRelNests ?elt2) (?n2 ifc:relatedObjects_IfcRelNests
?list2) (?p2 list:isIn ?list2)]

[sotHasElements-IFC: (?s sot:hasElements ?elt) <-
 (?c ifc:relatingGroup_IfcRelAssignsToGroup ?s)
 (?c ifc:relatedObjects_IfcRelAssigns ?elt)
 (?s rdf:type ?cls) (?cls b4r:subClassOf sot:System)
 (?elt rdf:type ?clse) (?clse b4r:subClassOf sot:SystemComponent)]


[ifcToLbdAirTerminalBox-USERDEFINED:(?z rdf:type ifc:IfcAirTerminalBox) (?z
ifc:predefinedType_AirTerminalBox ifc:USERDEFINED) -> (?z rdf:type
distribution:AirTerminalBox-USERDEFINED) .
[ifcToLbdAirTerminalBox-VARIABLEFLOWPRESSUREDEPENDANT:(?z rdf:type
ifc:IfcAirTerminalBox) (?z ifc:predefinedType_AirTerminalBox
ifc:VARIABLEFLOWPRESSUREDEPENDANT) -> (?z rdf:type distribution:AirTerminalBox-
VARIABLEFLOWPRESSUREDEPENDANT) .
[ifcToLbdAirTerminalBox-VARIABLEFLOWPRESSUREINDEPENDANT:(?z rdf:type
ifc:IfcAirTerminalBox) (?z ifc:predefinedType_AirTerminalBox
ifc:VARIABLEFLOWPRESSUREINDEPENDANT) -> (?z rdf:type distribution:AirTerminalBox-
VARIABLEFLOWPRESSUREINDEPENDANT) .
[ifcToLbdAirTerminalBox-CONSTANTFLOW:(?z rdf:type ifc:IfcAirTerminalBox) (?z
ifc:predefinedType_AirTerminalBox ifc:CONSTANTFLOW) -> (?z rdf:type
distribution:AirTerminalBox-CONSTANTFLOW) .
[ifcToLbdAirTerminalBox-NOTDEFINED:(?z rdf:type ifc:IfcAirTerminalBox) (?z
ifc:predefinedType_AirTerminalBox ifc:NOTDEFINED) -> (?z rdf:type
distribution:AirTerminalBox-NOTDEFINED) .

[ifcToLbdValve-ANTIVACUUM:(?z rdf:type ifc:IfcValve) (?z ifc:predefinedType_Valve
ifc:ANTIVACUUM) -> (?z rdf:type distribution:Valve-ANTIVACUUM) .


….
```